PROJECT REPORT

ON

Smart water monitoring system

Submitted

In Partial Fulfilment of the Requirements

For The Award of the Degree In

BACHELOR OF TECHNOLOGY
IN
ELECTRONICS AND COMMUNICATION ENGINEERING

Under the able guidance of by:	**Submitted by:**
Dr. Davinder Singh Saini	**Sheetanshu Agarwal (101110)**
Associate professor	**Heramb Shukla (101112)**
Electronics and Communication Engineering	**Umang Rajput (101123)**

Certificate

TO WHOM IT MAY CONCERN

This is to certify that project titled

“Smart water monitoring system”

Submitted by:

- **Sheetanshu Agarwal**
- **Heramb Shukla**
- **Umang Rajput**

in partial fulfilment for the award of degree of Bachelor of Technology in Electronics and Communication Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Signature:

Dr. D.S Saini

Department of Electronics and Communication Engineering

Jaypee University of Information Technology

Waknaghat Solan 173234

Acknowledgment

We take this opportunity to express our deep sense of gratitude and regard to **Dr Davinder Singh Saini**, Associate Professor, Department of Electronics and Communication Engineering, JUIT, Waknaghat for his continuous encouragement and able guidance, we needed to complete this project. We are highly obliged to him for providing us the opportunity for implementing our idea and helping us to complete it successfully.

We take immense pleasure in thanking **Prof. Sunil Bhooshan**, Head of the department of ECE for permitting us to carry out this project.

We express our deep sense of gratitude and sincere thanks to **Mrs. Meenakshi Sood**, Project coordinator, JUIT for giving us such an opportunity.

We are also highly obliged to **Mr Mohan Sharma**, Project Lab Coordinator, for his support and valuable encouragement throughout our project.

We are also grateful to our friend **Arpit Johari** of Civil Engineering who helped us in assembling the components.

Signature: Date: 29/05/2014

Sheetanshu Agarwal (101110)

Heramb Shukla (101112)

Umang Rajput (101123)

DECLARATION

We hereby declare that the work reported in B. tech report entitled **“SMART WATER MONITORING SYSTEM”,** submitted by **“Sheetanshu Agarwal, Heramb Shukla and Umang Rajput”** at Jaypee University of Information Technology, Waknaghat, Solan is an authentic record of or work carried out under the supervision of **Dr. Davinder Singh Saini**. This work has not been submitted partially or fully to any other university or institute for the award of this or any other degree or diploma

Signature: Date: 29/05/2014

Sheetanshu Agarwal (101110)

Heramb Shukla (101112)

Umang Rajput (101123)

Index

List of Figure

Summary

The project proposes a system designed to reduce water wastage in an organization. A limit on units of water consumed is assigned to each individual. The system monitors the volume of water used by every user and fine is charged accordingly if the limit is exceeded. Biometrics fingerprint recognition is used to identify the user and to maintain their respective accounts.

This project uses a flow-rate gauge that contains a Hall-effect sensor which outputs a series of pulses proportional to instantaneous flow rate that implement a simple frequency counter.

This system is a combination of both hardware as well as software aspects. The project uses Arduino environment that is a fusion of three critical elements: hardware, software and community. One needs to have a basic understanding of all three elements. It is a useful project in its own right and has various applications as it demonstrates a very useful technique that can be used in a wide range of projects that need to measure the rate at which something happens or to measure the volume of water like filling a bath, running an irrigation system providing adequate water supply to each crop as per the requirement.

Chapter1

Introduction

1.1 Introduction

Nowadays with rapidly increasing competition there is a great need of using existing water supplies efficiently thus, reducing water wastage and decreasing cost.

- **Motivation**: Increasing demand and decreasing supply of water as well as the problems faced by the college administration with the limited and difficult access to water supply and the cost associated with it, motivated us to do this project.
- **Objective:** This project aims at designing a system to monitor the use of water by an individual by assigning certain limit of water supply to every individual for a certain period of time and then charging a fine accordingly when the limit is exceeded. This limited use of water helps in reducing water wastage

1.2 Inspiration

- Public concepts of sharing and managing the finite supplies of water are changing. There exists an increasing competition between power, irrigation etc. Water districts need to find ways to extend the use of their shares of water using best available technologies. Best management measures and practices depend upon conservation of water and the key to conservation is good water measurement practices.
- As the demand of water increases, plans will be formulated to extend the use of water. Instead of finding and developing new sources, water can be less expensively provided by conservation and equitable distribution of existing water supplies. Every cubic foot of water recovered as a result of improved water measurement produces more revenue than the same amount obtained from a new source.

- More careful use of water by each individual reduces water wastage. Measurement of water supply is required by farmers for proper irrigation so that different crops are supplied with the water supply per the demand thus preventing reduced yields and other crop damage caused by over-watering.

1.3 Benefits

- Accurate accounting and good records help allocate equitable shares of water within farm, resulting in fewer problems and easier operation.
- Accounting for individual water combined with pricing policies which penalize excessive use results in efficient use of existing water sources by reducing water wastage**.**
- Good water measurement and management practices prevent excess runoff and percolation, that can cause damage to crops, pollute ground water with various chemicals and pesticides

Chapter 2

TECHNICAL DETAILS

2.1 COMPONENTS USED

- Flow sensor
- Arduino UNO
- Bread Board
- LCD (16x2)
- Connecting wires
- Variable resistor
- Pipe

2.1.1 Water flow sensor

Water Flow Sensor is the latest sensor, which mainly consisting of

- Plastic valve
- Water flow rotor parts
- Hall sensor

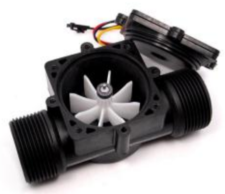


Fig 2.1 Internal design of sensor

Water Flow Sensor has the following features:

- Light and agile appearance, small, and simple installation.
- Stainless steel ball embedded in the impeller, lasting and wearing.
- The sealing ring adopting the on and under stress structure, no water leakage.
- Widely used in various controllers and Development Kit, like controllers(Arduino-Compatible), STC Mono-Chip computers and AVR Mono-Chip computers.

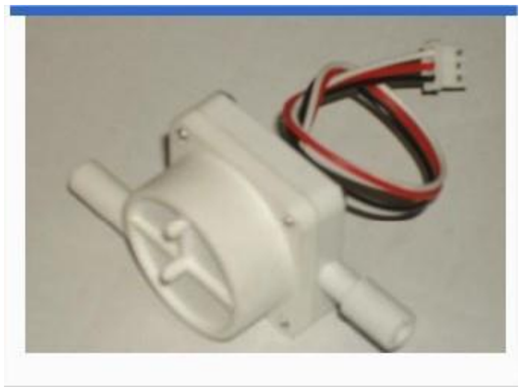


Fig 2.2 Flow Sensor

2.1.1(a) Operating Principle of Hall Effect sensor

To move the rotor in the commanded direction, the drive will send current through two of the motor's stator coils. This current produces electromagnetic fields that develop a torque on the rotor, and the rotor turns. The rotor will stop if it can reach a position where its permanent magnets are next to the magnetic fields that attract them. Before the rotor can get to this position, though, the drive switches the current to a new combination of stator coils, and creates a new set of electromagnetic field continue its motion.

The process of continually switching current to different motor coils to produce torque on the rotor is called commutation.

If the drive knows the position of the rotor's permanent magnets, it can set up magnetic fields in the stator that have the correct location and polarity to cause the rotor to turn.

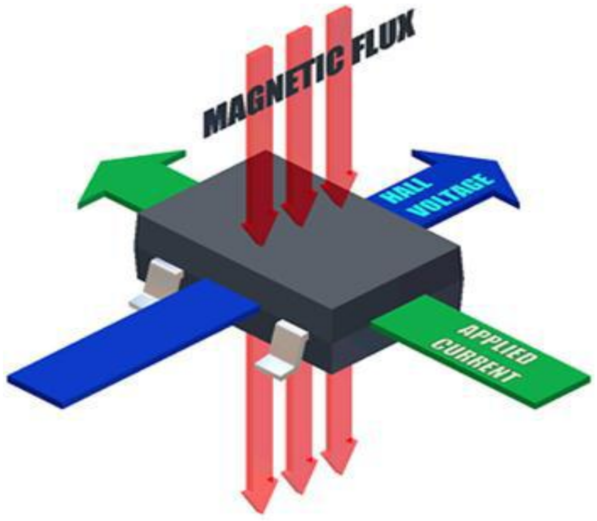


Fig 2.3 Hall effect principle

2.1.2 Arduino

Arduino provides an open source electronics prototyping platforms based on flexible and easy-to-use hardware and software. Arduino platforms are intended for designers, hobbyists, and anyone interested in creating interactive objects or environments. The prototyping platforms of Arduino sense the environment by receiving input from various sensors and can affect their surroundings by controlling lights, motors and other actuators. Arduino projects can be stand-alone or they can communicate with software running on a computer.

We have used Arduino UNO microcontroller having Atmel's ATmega328 series chip with built in USB. It has many advantages. It also offers more digital and analog pins.

The pulses coming from the output of the sensor are given to Arduino pin number 5. And the programming is done on the computer using Arduino software. Arduino counts these pulses and the volume is displayed on the serial monitor which is then displayed on LCD.

2.1.2 (a) Hardware description

Features

- Input voltage - 7-12V
- ATmega328 microcontroller
- 14 Digital I/O Pins (6 PWM outputs)
- 6 Analog Inputs
- 32k Flash Memory
- 16MHz Clock Speed

The maximum values that Arduino can handle

Max frequency: 16MHz

Max Voltage: 5V

Max Current: 50mA

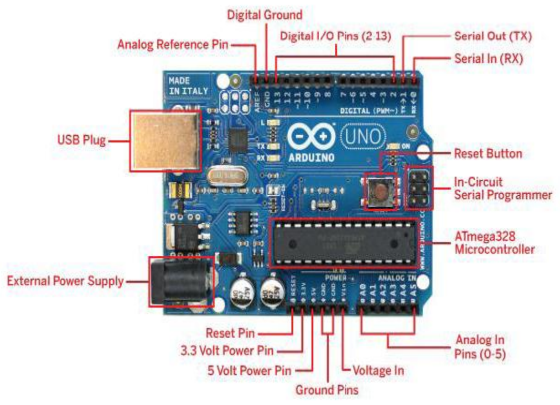


Figure 2.4 Arduino

Pin Description

- Arduino can be powered using power jack or USB port. It can also be powered using an external battery or AC to DC adaptor through pin Vin 5V, 3.3V: there is an inbuilt regulator on the board.
- Reset: This helps to reset the micro controller.
- IOREF: This pin acts as a reference to the inputs given to the Arduino board.
- There are 6 pins A0-A5 through which analog inputs can be given to the Arduino board.
- There are 14 digital pins 0-13. Out of these (3, 5, 6, 9, 10, 11) are PWM pins (pulse width modulation) from which analog output can be taken from the Arduino board.
- There is an inbuilt LED on pin 13.
- Rx, TX: Used to receive and transmit serial data.
- AREF: Acts as reference to the analog inputs.
- ICSP: (In Circuit Serial Programming) These pins enable the user to program the chips on the circuit

2.1.2(b) Software

The Arduino IDE is a cross-platform application written in Java, and is derived from IDE for Processing programming language and the wiring project. It is designed to introduce programming to newcomers unfamiliar with software development. It includes a code editor having features like syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. There is typically no need of editing makefiles or run programs on a command-line interface. Although building on command-line is possible I required with third party tools like Ino.

Arduino IDE comes with a C/C++ library called “Wiring” that makes many input/output operations much easier. Arduino programs are written in C/C++, although users only need to define two functions in order to make a runnable program:

- setup() – a function run once at the start of a program which can initialize settings.
- loop() – a function called repeatedly until the board powers off.

Most of the Arduino boards have an LED and load resistor connected between pin 13 and ground, a convenient feature for many simple tests. The above code would not be valid to a standard C++ compiler, so when user clicks the "Upload to I/O board" button in the IDE, a copy of the code is written to a temporary file with an extra included header at the top and a simple main() function at the bottom, in order to make it a C++ program. Arduino IDE uses GNU toolchain and AVR Libc for compiling programs and uses avrdude to upload programs to the board.

As the Arduino platform uses Atmel microcontrollers Atmel's development environment, AVR Studio or the newest Atmel Studio, may also be used for developing software for the Arduino. Its hardware reference designs are distributed under a Creative Commons Attribution Share-Alike 2.5 license and are available on the Arduino Web site. Layout and production files for some versions of Arduino hardware are also available. The source code for IDE and on board library are available and released under the GPLv2 license.

2.1.3 LCD Display 16*2

LCD (Liquid Crystal Display) screen is an electronic display module and finds a wide range of applications.

LCDs are economical and easily programmable.

A **16x2 LCD** means it can display 16 characters per line and there are 2 such lines.

Each character is displayed in 5x7 pixel matrix

Fig 2.5 LCD 16x2

2.1.4 Fingerprint Module

Theory

Fingerprint

Fingerprint in its narrow sense is an impression left by the friction ridges of a human finger. In a wider use of the term, fingerprints are the traces of an impression from the friction ridges of any part of a human. A print from the foot can also leave impression of friction ridges. A friction ridge is a raised portion of the epidermis on the digits (fingers and toes), the palm of the hand or the sole of the foot, that consists of one or more connected ridge units of friction ridge skin. These are sometimes called as "epidermal ridges" which are caused by the underlying interface between the dermal papillae of the dermis and the interpapillary (rete) pegs of the epidermis. These epidermal ridges serve to amplify the vibrations triggered, for example, when fingertips brush across an uneven surface, better transmitting the signals to sensory nerves involved in fine texture perception. These ridges can also help in gripping rough surfaces and may improve surface contact in wet conditions.

Fingerprint impressions may be left behind on a surface by the natural secretions of sweat from the Eccrine glands which are present in friction ridge skin, or they may be made by ink or other substances transferred from the peaks of friction ridges on the skin to a relatively smooth surface such as a fingerprint card. Fingerprint records normally contain impressions from the pad on last joint of fingers and thumbs, although fingerprint cards also typically record portions of lower joint areas of the fingers.

Fingerprint identification or recognition

Fingerprint identification or recognition, known as dactyloscopy, or hand print identification, is the process of comparing two instances of friction ridge skin impressions from human fingers or toes, or even the palm of the hand or sole of the foot, in order to determine whether these impressions could have come from the same individual. Fingerprint authentication is the automated method of verifying a match between two human fingerprints. Fingerprints are one of various forms of biometrics used to identify individuals and verify their identity. The flexibility of friction ridge skin means that no two finger or palm prints are ever exactly alike in every detail; even two impressions recorded immediately after each other from the same hand may differ slightly. Fingerprint identification, also referred to as individualization, involves an expert, or an expert computer system operating under threshold scoring rules, determining if two friction ridge impressions are likely to have originated from the same finger or palm (or toe or sole).

The analysis of fingerprints for matching purposes generally requires the comparison of various features of the print pattern. These consist of patterns, which are aggregate characteristics of ridges, and minutia points, which are unique features found within the patterns. It is also essential to know the structure and properties of human skin in order to successfully employ some of the imaging technologies.

The three basic patterns of fingerprint ridges are arch, loop, and whorl:

- Arch: The ridges enter from one side of the finger, rise in the center forming an arc, and then exit the other side of the finger.
- Loop: The ridges enter from one side of a finger, form a curve, and then exit on that same side.
- Whorl: Ridges form circularly around a central point on the finger.

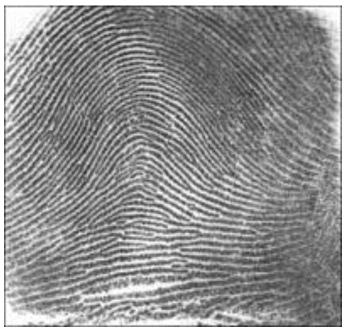


Fig 2.6 The arch pattern

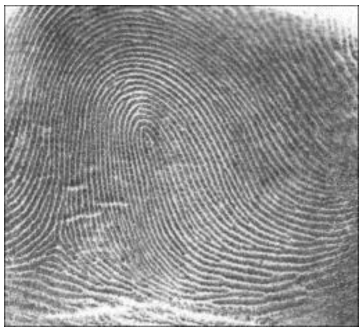


Fig 2.7 The loop pattern

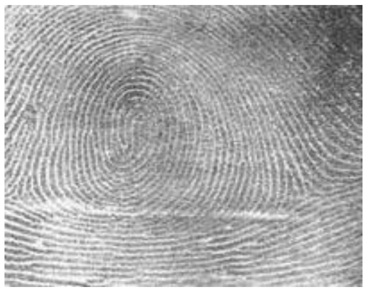


Fig 2.8 The whorl pattern

When friction ridges come in contact with a surface that will take a print, material that is on the friction ridges such as perspiration, oil, grease, ink or blood, will be transferred to the surface. There are numerous factors which affect the quality of friction ridge impressions. Pliability of the skin, deposition pressure, slippage, the material from which the surface is made, the roughness of the surface and the substance deposited are some of the various factors which can cause a latent print to appear differently from any known recording of the same friction ridges. The conditions surrounding every instance of friction ridge deposition are unique and never duplicated. For such reasons, fingerprint examiners are required to undergo extensive training.

Minutia features

In biometrics and forensic science, **minutiae** are major features of a fingerprint, using which comparisons of one print with another can be made.

Ridge ending, bifurcation, and short ridge (or dot) are the major minutia features of fingerprint ridges.

- Ridge ending: The ridge ending is the point at which a ridge terminates.
- Bifurcation: Bifurcation is the point at which a single ridge splits into two ridges.
- Short ridges (or dots): Short ridges are the ridges which are significantly shorter than the average ridge length on the fingerprint.

Minutiae and patterns are very important in the analysis of fingerprints as no two fingers have been shown to be identical.

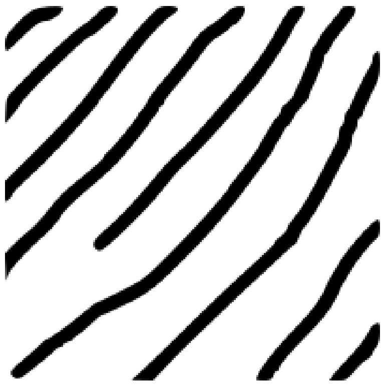


Fig 2.9 Ridge ending

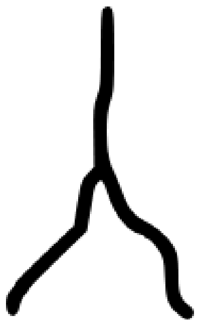


Fig 2.10 Bifurcation

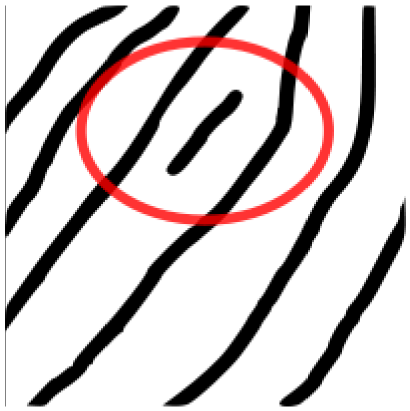


Fig 2.11 Short ridge (dot)

Fingerprint sensors

A fingerprint sensor is an electronic device which is used to capture a digital image of the fingerprint pattern. The captured image is known as a live scan. This live scan is digitally processed to create a biometric template (a collection of extracted features) that is stored and used for matching. Here is an overview of some of the more commonly used fingerprint sensor technologies.

Optical: Optical fingerprint imaging involves capturing a digital image of the print using visible light. Optical sensor is, in essence, a specialized digital camera. The top layer of the sensor, where the finger is placed, is called the touch surface. Beneath this layer is a light-emitting phosphor layer that illuminates the surface of the finger. The light reflected from the finger passes through the phosphor layer to an array of solid state pixels (a charge-coupled device) that captures a visual image of the fingerprint. A scratched or dirty touch surface can cause a bad image of the fingerprint. A disadvantage of this type of sensor is that the imaging capabilities are affected by the quality of skin on the finger. For instance, it is difficult to take the image of a dirty or marked finger properly. It is also possible for an individual to erode the outer layer of skin on the fingertips to the point where the fingerprint is no longer visible. It may also be easily fooled by an image of a fingerprint if not coupled with a "live finger" detector. However, unlike capacitive sensors, optical sensor technology is not susceptible to electrostatic discharge damage.

Ultrasonic: Ultrasonic sensors use the principles of medical ultrasonography in order to create visual images of the fingerprint. Ultrasonic sensors use very high frequency sound waves to penetrate the epidermal layer of skin. The sound waves are generated using piezoelectric transducers and the reflected energy is also measured using piezoelectric materials. Because the dermal skin layer exhibits the same characteristic pattern of the fingerprint, the reflected wave measurements can be used to form an image of the fingerprint. This eliminates the need for clean, undamaged epidermal skin and a clean sensing surface.

Capacitance: Capacitance sensors use principles associated with capacitance to form fingerprint images. The sensor array pixels each act as one plate of a parallel-plate capacitor, the dermal layer (which is electrically conductive) acts as the other plate, and the non-conductive epidermal layer acts as a dielectric.

Passive capacitance: A passive capacitance sensor uses the principle outlined above to form an image of the fingerprint patterns on the dermal layer of the skin. Every sensor pixel is used to measure the capacitance at that point of the array. The capacitance varies between the ridges and valleys of the fingerprint because of the fact that the volume between the dermal layer and sensing element in valleys contains an air gap. The values of dielectric constant of the epidermis and the area of the sensing element are known. The measured capacitance values are then used for distinguishing between fingerprint ridges and valleys.

Active capacitance: Active capacitance sensors use a charging cycle for applying a voltage to the skin before measurement takes place. This voltage charges the effective capacitor. The electric field between the finger and the sensor follows the pattern of the ridges in the dermal skin layer. On the discharge cycle, the voltage across the dermal layer and sensing element is compared against a reference voltage in order to calculate the capacitance. The distance values are then calculated mathematically, and used to form an image of the fingerprint. Active capacitance sensors measure the ridge patterns of the dermal layer like the ultrasonic method. Again, this eliminates the need for clean, undamaged epidermal skin and a clean sensing surface.

Algorithms

Matching algorithms are used for comparing previously stored templates of fingerprints against candidate fingerprints for authentication purposes. In order to do this either the original image must be directly compared with the candidate image or certain features must be compared.

Pattern-based (or image-based) algorithms

Pattern based algorithms compare the basic fingerprint patterns (arch, whorl, and loop) between a previously stored template and a candidate fingerprint. This requires that the images be aligned in the same orientation. To do this, the algorithm finds a central point in the fingerprint image and centres on that. In a pattern-based algorithm, the template contains the type, size, and orientation of patterns within the aligned fingerprint image. The candidate fingerprint image is graphically compared with the template to determine the degree to which they match.

Feature extraction

In pattern recognition, **feature extraction** is a special form of dimensionality reduction. When the input data to an algorithm is too large to be processed and it is suspected to be notoriously redundant (e.g. the same measurement in both feet and meters) then the input data is transformed into a reduced representation set of features (also named features vector). Transforming the input data into the set of features is called feature extraction. If the features extracted are carefully chosen it is expected that the features set will extract the relevant information from the input data in order to perform the desired task using this reduced representation instead of the full size input.

Feature extraction involves simplifying the amount of resources required to describe a large set of data accurately. When performing analysis of complex data one of the major problems stems from the number of variables involved. Analysis with a large number of variables generally requires a large amount of memory and computation power or a classification algorithm which overfits the training sample and generalizes poorly to new samples. Feature extraction is a general term for methods of constructing combinations of the variables to get around these problems while still describing the data with sufficient accuracy.

Best results are achieved when an expert constructs a set of application-dependent features. Nevertheless, if no such expert knowledge is available general dimensionality reduction techniques may help. One such technique is **Principal component analysis** (**PCA**) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called **principal components**. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to (i.e., uncorrelated with) the preceding components. Principal components are guaranteed to be independent if the data set is jointly normally distributed. PCA is sensitive to the relative scaling of the original variables.

In this project we have used R30X Series Fingerprint Identification Module. We chose this sensor because it is easy to use and it comes with fairly straight-forward Windows software that makes the entire process simple.

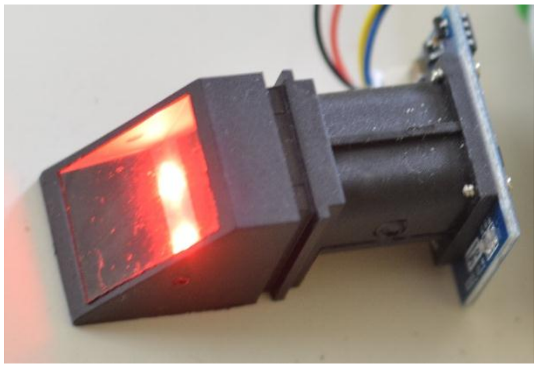


Fig 2.12 R305 fingerprint sensor module

Specifications

POWER	DC 3.6V- 6.0V	INTERFACE	UART(TTL LOGICAL LEVEL)
Working current	Typical: 100mA Peak: 150mA	Matching Mode	1:1 and 1:N
Baud rate	(9600*N)bps, N=1~12 (default N=6)	Character file size	256 bytes
Image acquiring time	<0.5s	Template size	512 bytes
Storage capacity	256	Security level	5(1,2, 3, 4, 5(highest))

Table 1 Specification of finger print module

Operating Principle

Fingerprint processing includes two parts: fingerprint enrollment and fingerprint matching (matching can be 1:1 or 1: N).

While enrolling, the user needs to enter the finger two times. System will process the two time finger images, generate a template of the finger based on processing results and store the template.

While matching, user needs to enter the finger image through optical sensor and system will generate a template of the finger and compare it with the templates of the finger library. For 1:1 matching, system compares the live finger with specific template designated in the module; for 1:N matching, or searching, system will search the whole finger library for the matching finger. In both cases, system will return the matching result, success or failure.

Pin Description

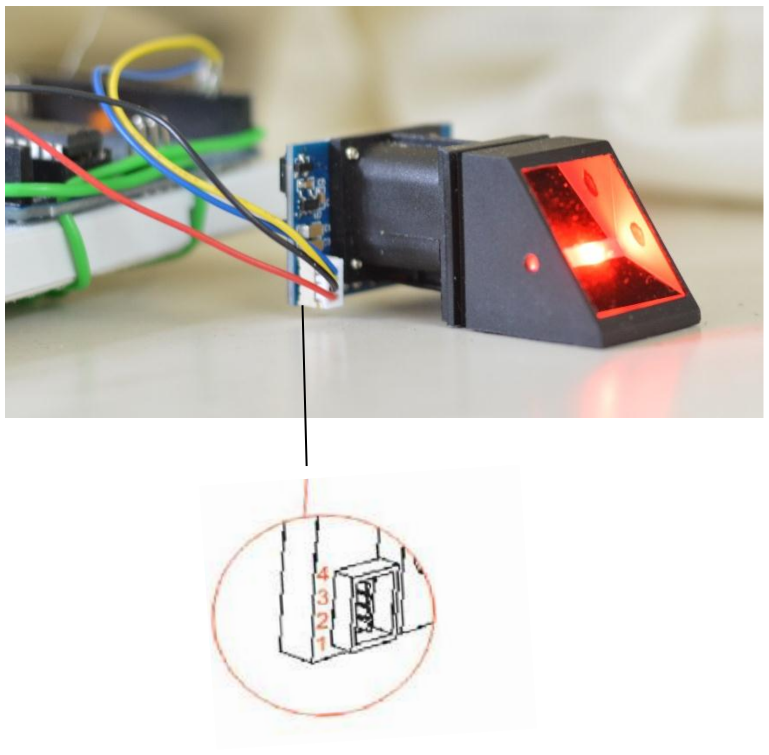


Fig 2.13 Fingerprint module pin description

Serial Communication (P1)

Pin Number	**Name**	**Type**	**Function Description**
1	Vin	in	Power input
2	GND	_	Signal ground. Connected to power ground (color: black)
3	TD	In	Data output. TTL logical level
4	RD	Out	Data input. TTL logical level

Table 2 Pin description of finger print module

Hardware connection

Via serial interface, the Module can communicate with MCU of 3.3V or 5V power. TD (pin 3 of P1) connects with RXD (receiving pin of MCU), RD (pin 4 of P1) connects with TXD (transferring pin of MCU).

System Resources

To meet the demands of different customers, Module system provides abundant resources at user's use.

NOTEPAD

The system sets aside a 512-bytes memory for user's notepad, where data that requires power-off protection can be stored. The page can be accessed by the host using instructions of PS_WriteNotepad and PS_ReadNotepad.

BUFFER

There are 3 buffers: an image buffer and two 512-byte-character-file buffer within the RAM space of the module. Users can read and write any of the buffers using instructions. (Contents of the buffers will be lost at power off.)

- Image buffer : ImageBuffer is used for image storage and the image format is 256*288 pixels.
- Character file buffer : Character file buffer, CharBuffer1, CharBuffer2, is used to store both character file and template file.

Fingerprint library

System sets aside a certain space within Flash for fingerprint template storage called fingerprint library. Contents of the library remain at power off.

Capacity of the library changes with capacity of Flash, system will recognize the latter automatically. Storage of fingerprint template in flash is in sequential order. Assuming the fingerprint capacity to be N, then the serial number of template in library is 0, 1, 2, 3…N. User can only access library by the template number.

Software

There are basically two requirements for using the optical fingerprint sensor. First to enroll fingerprints – that means assigning ID #'s to each print so one can query them later. Once all the prints have been enrolled, one can easily search the sensor, asking it to identify which ID (if any) is currently being photographed.

Various steps involved are:

Place the finger on the fingerprint sensor after an ID has been assigned. Then when asked again place the same finger so that it can be verified before it is enrolled.

The 'confidence' is a score number (from 0 to 255) which indicates how good of a match the print is, higher is better. Note that if it matches at all, that means the sensor is pretty confident so One doesn’t have to pay attention to the confidence number unless it makes sense for high security applications.

Chapter 3

METHODOLOGY

3.1 INTRODUCTION

This chapter will cover the details explanation theory of methodology that is being used to make this project complete and working well. Many methodology or findings from this project mainly generated into project of same kind for others to take advantages and improve as upcoming studies. The method is used to achieve the objective of the project that will accomplish a perfect result.

maintenance and support	Analysis	implementation	design	Planning

Fig 3.1 SLDC Phase

This final year project used three major steps to implement project starting from planning, implementing and testing the part done.

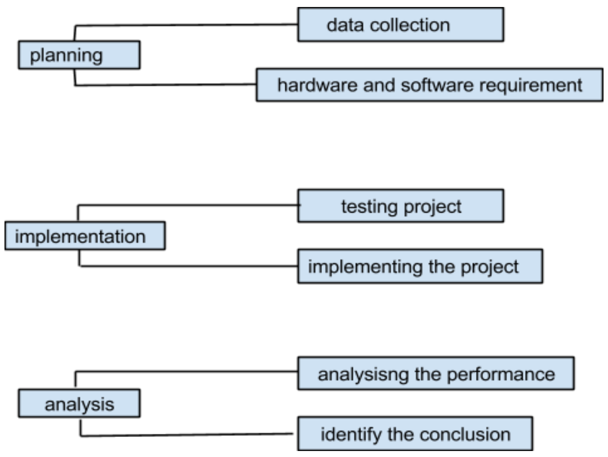


Fig 3.2 Steps of Methodology

3.2 PLANNING

To identify all the information and requirements such as hardware and software, planning is done in the proper manner. The planning phase consisted two main elements namely data collection and the requirements of hardware and software.

3.2.1 Data collection

At this stage we planned about the project's resources and requirements, literature studies and schedule and the manner in which the project proceeds. All the materials were collected from journal, texts book and blogs gathered from libraries and Internet.

Within the data collection period we studied in a detail manner about the Arduino board and Hall Effect sensor on the Internet and did research about the project. Once we were done with the literature study, we tried to find out the accessibility the electronic components and other materials to be used in project in the Indian market so that it does not hinder the progress in the future.

While planning, we did the research about the project, which includes study about the electronic components used such as Arduino, hall-effect sensor, finger print module and values of various resistors attached to it. The study is not just about the functions of various devices but also their working by attaching them in independent circuits to understand there working effectively.

3.2.2 Hardware Requirement

Below is the list of the entire electronic components and the other materials required.

- Flow sensor
- Arduino UNO
- Bread Board
- LCD (16x2)
- Connecting wires and resistor
- Finger print module
- Pipe

3.2.3 Software Requirement

Arduino software

Installing the software

Using access of the internet, there are step-by-step directions and procedure about the software available at: http://Arduino.cc/en/Main/Software or using the USB stick in the kit2 has the software under the Software Directory, it can be downloaded from there. There are two directories under that one is for "Windows" and the other is "Mac OS X". If you are installing onto Linux, you will need to follow the directions at: http://Arduino.cc/en/Main/Software or you can use the terminal.

Arduino software sketch

Basically Arduino sketch consists of two main functions to write a program which are namely

- Void setup()
- Void loop()

Void setup ():

Setup () is called when a sketch starts. It is used to initialize variables, pin modes, start using libraries etc. The setup () will only run once, after each power up or reset of the Arduino board.

Syntax:

```
Void setup ()

{

Statements;

}
```

Void loop ():

After creating a setup () function which initializes and set up the initial values, the loop () function is executed and it does precisely what its name suggests, and loops consecutively, allowing your program to respond and change. It is used to actively control the Arduino board.

Syntax:

```
Void loop ()
{
Statements;
}
```

Below we show a demonstration of a simple code to light a led using Arduino.

```
#define LED_PIN 13
 void setup () {
  pinMode (LED_PIN, OUTPUT); // It enable pin 13 for digital output
}
 void loop () {
  digitalWrite (LED_PIN, HIGH); // it turn on the LED
  delay (1000); // Wait one second (1000 milliseconds)
  digitalWrite (LED_PIN, LOW); // Turn off the LED
  delay (1000); // Wait one second
}
```

3.3 IMPLEMENTATION

3.3.1 Construction

- Mounting component parts on the breadboard, we used the parts-placement diagram given on the Arduino help page as a guide.
- Install the resistors finger print module and components on the board.
- The next step is to grip the wires in position and do all the necessary connections.
- The dc power supply used by Arduino is supplied by the computer and the supply to the Hall Effect sensor, LCD display and the finger print module are provided by the Arduino external supply given in the board

LCD 3.3V

Hall Effect sensor 5V

Fingerprint module 5V

After verifying all the connections next step is to solder the circuit to keep it intact.

3.3.2 Checking

After mounting the components on breadboard, we need to check the continuity track of the circuit. This part of the job is to ensure that the operation of this circuit will run smoothly.

The tools related with the checking part are multimeter and the continuity checking involves every circuit track and the points of soldering. By using the multimeter, it will alert the failed continuity. The failed continuity will recover with the solder again the lack of components related.

3.3.3 Testing

After the process of designing the circuit, the simulation of the circuit process is followed. By using the same software Arduino Integrated Development Environment the result of simulation was performed. Several simulation characteristics that can be observed by using this software are calculated such as the calibration factor possible for the Hall Effect sensor, the enrolment of the finger print module , the verification of the module so that each of the operation is perfectly synchronize with each other.

Determining the Calibration Factor

A bucket and a stopwatch are used for determining the calibration factor. The stopwatch is started when the flow starts, and stopped when the bucket reaches its limit say 1litre. The volume divided by the time gives the flow rate. For continuous measurements, we need a system of continually filling and emptying buckets to divide the flow without letting it out of the pipe. These continuously forming and collapsing volumetric displacements may take the form of pistons reciprocating in cylinders, gear teeth mating against the internal wall of a meter or through a progressive cavity created by rotating oval gears or a helical screw.

1 litre = 102 pulses on Hall Effect sensor which takes 40 sec (2/3 min)

1 litre/ min will be 102/ (2/3) =153 pulses/min or 2.55 pulse/sec

This means our scaling factor to convert pulses per second into litres per minute is 2.55.. By measuring the pulse frequency and dividing by 2.55 we can determine the current flow rate in litres per minute.

Code

```
#include<FreqCount.h>  // header file to count the number of pulses
float flowrate=0;
float volps=0;
float Total_vol=0;
void setup()
{
  Serial.begin(9600); //9600=baud rate
  FreqCount.begin(1000);  //no. of pulses is counted in each 1000ms
}
void loop()
{
  if(FreqCount.available())
  {
    unsigned long pulse=FreqCount.read(); //stores the number of pulses in 1000ms in variable pulse
    volps=pulse * .39; //.39 is our  calliberation factor
    Total_vol= Total_vol + volps;
    Serial.print("No. of Pulse=");
    Serial.println(pulse);
    Serial.print(" Volume(mL)=");
    Serial.println(Total_vol);
```

}

}

LCD display

LCD displays are most of the times driven using an industrial standard established by Hitachi (Explained in detail earlier). According to it there is a group of pins dedicated to sending data and locations of that data on the screen, the user can choose to use 4 or 8 pins to send data. On top of that three more pins are needed to synchronize the communication towards the display. We control Liquid Crystal Display (LCD) using the Arduino Liquid Crystal library. The library provides functions for accessing any LCD currently implements 8-bit control and one line display of 16x2 characters. Functions are provided to initialize the screen, to print characters and strings, to clear the screen, and to send commands directly to the HD44780 chip.

The LCDs have a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The interface consists of the following pins:

- A register select (RS) pin that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.
- A Read/Write (R/W) pin that selects reading mode or writing mode
- An Enable pin that enables writing to the registers
- 8 data pins (D0 -D7). The states of these pins (high or low) are the bits that you're writing to a register when you write, or the values you're reading when you read.
- There's also a display constrast pin (Vo), power supply pins (+5V and Gnd) and LED Backlight (Bklt+ and BKlt-) pins that you can use to power the LCD, control the display contrast, and turn on and off the LED backlight, respectively.

The process of controlling the display involves putting the data that form the image of what you want to display into the data registers, then putting instructions in the instruction register. The Liquid_Crystal Library simplifies this for you so you don't need to know the low-level instructions.

The Hitachi-compatible LCDs can be controlled in two modes: 4-bit or 8-bit. The 4-bit mode requires seven I/O pins from the Arduino, while the 8-bit mode requires 11 pins. For displaying text on the screen, you can do most everything in 4-bit mode, so example shows how to control a 2x16 LCD in 4-bit mode**.**

```
#include<FreqCount.h>  // header file to count the number of pulses
#include<LiquidCrystal.h>
float flowrate=0;
float volps=0;
float Total_vol=0;
LiquidCrystal lcd(11, 12, 6, 4, 3, 2);
void setup()
{
  Serial.begin(9600); //9600=baud rate
  FreqCount.begin(1000);  //no. of pulses is counted in each 1000ms
  lcd.begin(16,2);


}


lcd.print(1000-Total_vol);
    }
    else
    {lcd.setCursor(0,0);
    lcd.print("Ex Vol(ml) =");
    lcd.print(Total_vol-1000);
    lcd.setCursor(0,1);
    lcd.print("Fine: Rs ");
    lcd.print((Total_vol-1000)*0.1);
    }
  }
```

The above program delivers the output to the lcd port. The output after calculating all the desired outputs is interfaced to lcd where all the values are displayed.

Final code

```
#include<FreqCount.h>  // header file to count the number of pulses
#include<LiquidCrystal.h>
float flowrate=0;
float volps=0;
float Total_vol=0;
LiquidCrystal lcd(11, 12, 6, 4, 3, 2);
void setup()
{
  Serial.begin(9600); //9600=baud rate
  FreqCount.begin(1000);  //no. of pulses is counted in each 1000ms
  lcd.begin(16,2);

}
void loop()
{
  if(FreqCount.available())
  {
    unsigned long pulse=FreqCount.read(); //stores the number of pulses in 1000ms in variable pulse
    volps=pulse * .39; //.39 is our  calliberation factor
    Total_vol= Total_vol + volps;
    Serial.print("No. of Pulse=");
    Serial.println(pulse);
    Serial.print(" Volume(mL)=");
    Serial.println(Total_vol);
    if(Total_vol<= 1000)
    { lcd.setCursor(0,0);
    lcd.print("Water left ");
    lcd.setCursor(0,1);
    lcd.print("Vol(ml)=");

    lcd.print(1000-Total_vol);
    }
    else
    {lcd.setCursor(0,0);
    lcd.print("Ex Vol(ml) =");
    lcd.print(Total_vol-1000);
    lcd.setCursor(0,1);
    lcd.print("Fine: Rs ");
    lcd.print((Total_vol-1000)*0.1);
    } }}
```

Finger print module

There are basically two steps involved in the working of the fingerprint module with the Arduino microcontroller which are enrolment and verification. First you need to enrol the fingerprint and secondly it needs to be verified. Enrolling is assigning ID# to each print so you can query them later. Once you have enrolled then you can ask the sensor to identify the print of the Id used to verify it with the user data stored.

Enrolment

```
#include <rs_Fingerprint.h>
#if ARDUINO >= 100
 #include <SoftwareSerial.h>
#else
 #include <NewSoftSerial.h>
#endif

uint8_t getFingerprintEnroll(uint8_t id);



// pin #2 is IN from sensor (YELLOW wire)
// pin #3 is OUT from Arduino  (BLUE wire)
#if ARDUINO >= 100
SoftwareSerial mySerial(2, 3);
#else
NewSoftSerial mySerial(2, 3);
#endif

rs_Fingerprint finger = rs_Fingerprint(&mySerial);

void setup()
{
  Serial.begin(9600);
  Serial.println("fingertest");
  // set the data rate for the sensor serial port
  finger.begin(57600);

  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  } else {
    Serial.println("Did not find fingerprint sensor :(");
    while (1);
  }
}

void loop()                     // run over and over again
{
  Serial.println("Type in the ID # you want to save this finger as...");
  uint8_t id = 0;
  while (true) {
    while (! Serial.available());
    char c = Serial.read();
    if (! isdigit(c)) break;
    id *= 10;
    id += c - '0';
  }
  Serial.print("Enrolling ID #");
  Serial.println(id);

  while (!  getFingerprintEnroll(id) );
}
```

Verification

```
#include <Rs_Fingerprint.h>
#if ARDUINO >= 100
```

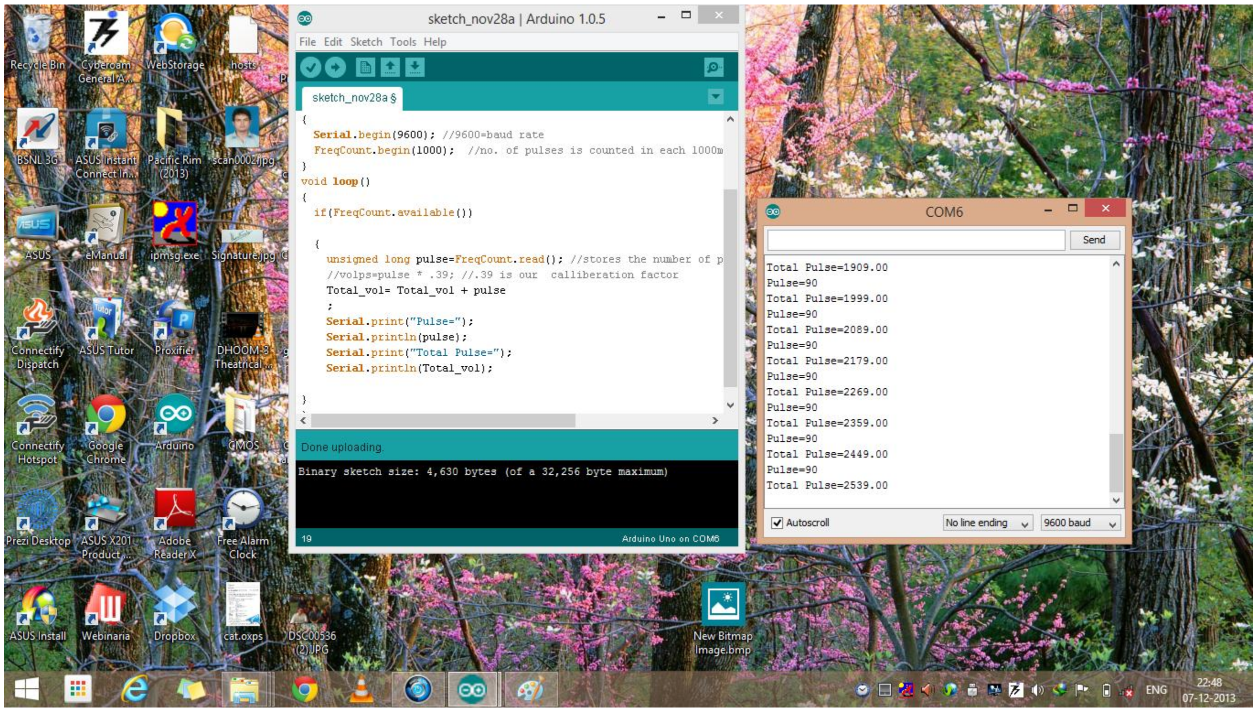
```
 #include <SoftwareSerial.h>
#else
 #include <NewSoftSerial.h>
#endif

int getFingerprintIDez();

// pin #2 is IN from sensor (GREEN wire)
// pin #3 is OUT from Arduino  (WHITE wire)
#if ARDUINO >= 100
SoftwareSerial mySerial(2, 3);
#else
NewSoftSerial mySerial(2, 3);
#endif

Rs_Fingerprint finger = Rs_Fingerprint(&mySerial);

void setup()
{
  Serial.begin(9600);
  Serial.println("fingertest");

  // set the data rate for the sensor serial port
  finger.begin(57600);

  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  } else {
    Serial.println("Did not find fingerprint sensor :(");
    while (1);
  }
  Serial.println("Waiting for valid finger...");
}

void loop()                    // run over and over again
{
  getFingerprintIDez();
}

int getFingerprintIDez() {
  uint8_t p = finger.getImage();
  if (p != FINGERPRINT_OK)  return -1;

  p = finger.image2Tz();
  if (p != FINGERPRINT_OK)  return -1;

  p = finger.fingerFastSearch();
  if (p != FINGERPRINT_OK)  return -1;

  // found a match!
  Serial.print("Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of "); Serial.println(finger.confidence);
  return finger.fingerID;
}
```

3.3.4 Installation

After the successful implementation of the desired result the final code is burned into the Arduino board and the output is analysed.

3.4 ANALYSIS

The analysis stage is the final stage in this methodology where two points will be done. The analysis is based on the performance of the circuit related where the output must be performed well and successful and the second is identifying the conclusion. The output of the circuit on the computer when the calibration factor was calculated:



```
{
  Serial.begin(9600); //9600=baud rate
  FreqCount.begin(1000);  //no. of pulses is counted in each 1000m
}
void loop()
{
  if(FreqCount.available())

  {
    unsigned long pulse=FreqCount.read(); //stores the number of p
    //volps=pulse * .39; //.39 is our  calliberation factor
    Total_vol= Total_vol + pulse
    ;
    Serial.print("Pulse=");
    Serial.println(pulse);
    Serial.print("Total Pulse=");
    Serial.println(Total_vol);

}
```

```
Done uploading.
Binary sketch size: 4,630 bytes (of a 32,256 byte maximum)
19                                         Arduino Uno on COM6
```

```
Total Pulse=1909.00
Pulse=90
Total Pulse=1999.00
Pulse=90
Total Pulse=2089.00
Pulse=90
Total Pulse=2179.00
Pulse=90
Total Pulse=2269.00
Pulse=90
Total Pulse=2359.00
Pulse=90
Total Pulse=2449.00
Pulse=90
Total Pulse=2539.00
```

Autoscroll [✓]   No line ending   9600 baud

Fig 3.3: Pulses generated by the Hall Effect sensor

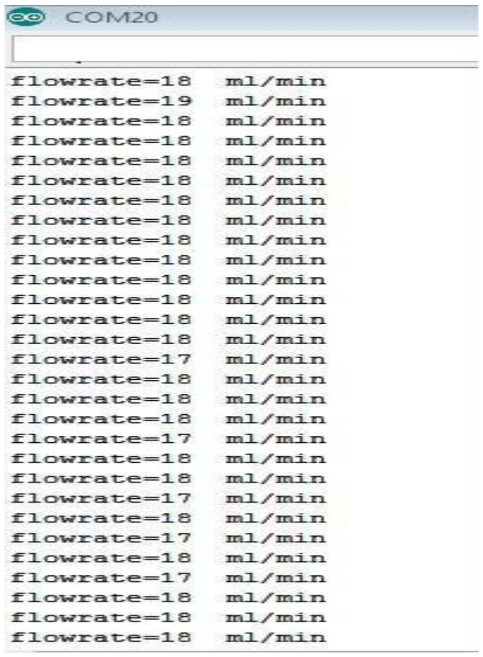
In the next we discussed all the result in great detail to show how these PWM pulse generated as water flows (shown in Fig 3.3) through the sensor are used to generate the water flowing across the sensor in ml. We will be showing the detailed analysis of how we accurate measured the water and also how identification system that we use in this project with the help of finger print module is integrated with the Arduino and Hall Effect sensor.

Chapter 4

Results

In the following chapter we discuss all the result individually and in a combined way. We discussed the interface of all the devices attached to Arduino and similarly show those result in various figure taken during the course of project development. We will look into results of Hall Effect sensor its pulses and then we calculate the water flow and then the identification using finger print module in which we will enroll and verify the user and then integrate the sensor and the finger print module. We will also use LCD so as to display all the result without the use of computer so that the independency of the project is maintained.

4.1 Observation on the serial monitor

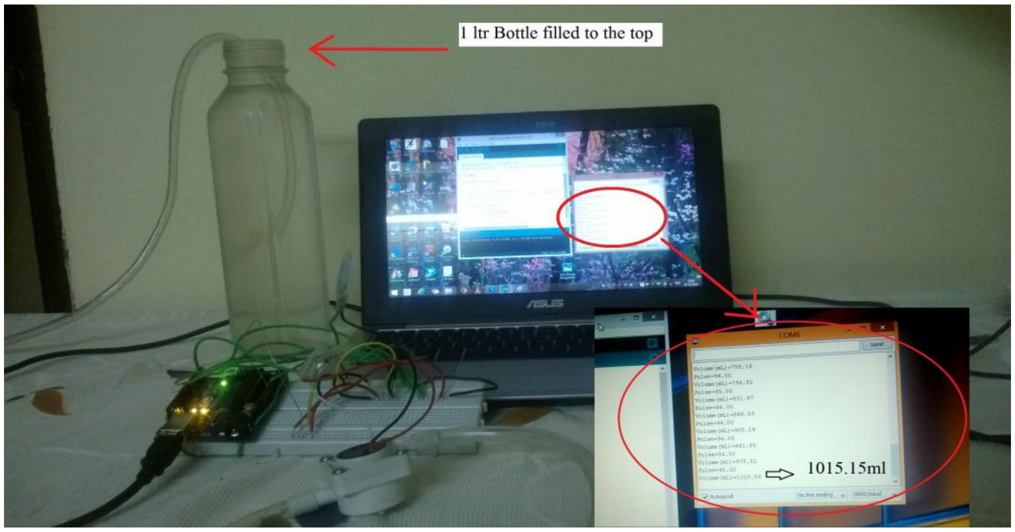


```
COM20

flowrate=18  ml/min
flowrate=19  ml/min
flowrate=18  ml/min
flowrate=18  ml/min
flowrate=18  ml/min
flowrate=18  ml/min
flowrate=18  ml/min
flowrate=18  ml/min
flowrate=18  ml/min
flowrate=18  ml/min
flowrate=18  ml/min
flowrate=18  ml/min
flowrate=18  ml/min
flowrate=18  ml/min
flowrate=17  ml/min
flowrate=18  ml/min
flowrate=18  ml/min
flowrate=18  ml/min
flowrate=17  ml/min
flowrate=18  ml/min
flowrate=18  ml/min
flowrate=17  ml/min
flowrate=18  ml/min
flowrate=17  ml/min
flowrate=18  ml/min
flowrate=17  ml/min
flowrate=18  ml/min
flowrate=18  ml/min
flowrate=18  ml/min
```

Fig 4.1 Observation of flow rate on serial monitor for different flow rates

We used the calibration factor to determine the amount of water flowing through the pipe, it is basically the conversion factor of how the PWM pulse is converted to water measuring system. We used the S. I.

measurement system and calculated the water flow in milliliters. As we can see in Fig 4.1 the amount of water flowing per minute is calculated using calibration factor and shown the displayed in serial monitor. We used this result in calculating the water by adding the value which will give the actual water flowing in real time. We can see in Fig 4.2 we have shown the total water that has been flowing through sensor and the value is accumulated and the result is very near to the actual value as we are using the 1 lit water bottle and when the bottle is filled we can see that correspondingly a value of 1015 ml is shown in the serial monitor.



```
Volume(mL)=758.16
Pulse=94.00
Volume(mL)=794.82
Pulse=95.00
Volume(mL)=831.87
Pulse=94.00
Volume(mL)=868.53
Pulse=94.00
Volume(mL)=905.19
Pulse=94.00
Volume(mL)=941.85
Pulse=94.00
Volume(mL)=978.51
Pulse=95.00
Volume(mL)=1015.56
```

Fig 4.2 Total water used in serial monitor

4.2 Observation on LCD

LCD attached to the Arduino give us the real time usage of the water without the use of serial monitor we have programed the LCD to display the amount of water allocated to the user and the after exceeding the limit how much fine he has to pay as a result of excessive use of water and both these are carried out and there result are shown below in terms of Fig 4.3 and Fig 4.4. Fig 4.3 shows the amount of water allocated to a particular user in ml, fig 4.4 shows the fine imposed on the user after he exceeds the limit which he has been allocated

4.2.1 Before the allotted limit is crossed

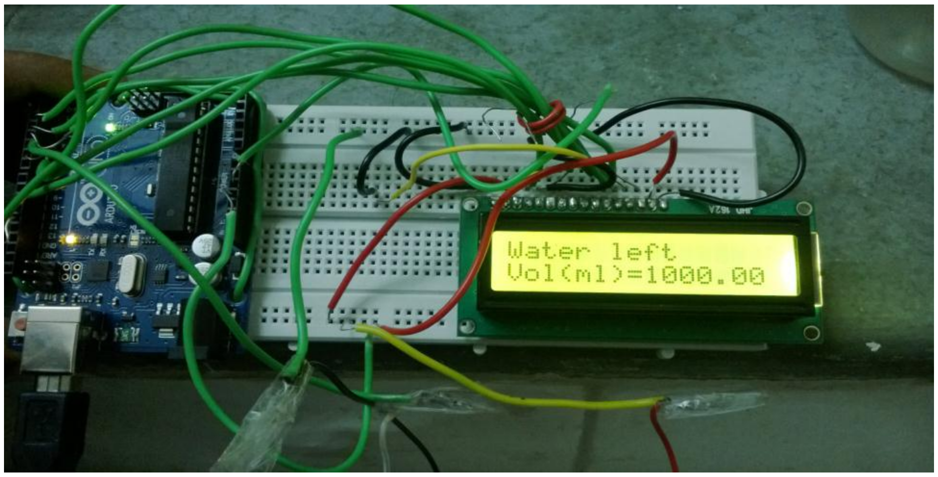


Fig 4.3 LCD displaying the water allotment

As we can see it shows the LCD which display the amount of water which the user is free to use and as when he exceeds the limit fine is impossed.

4.2.2 After the allotted limit is crossed

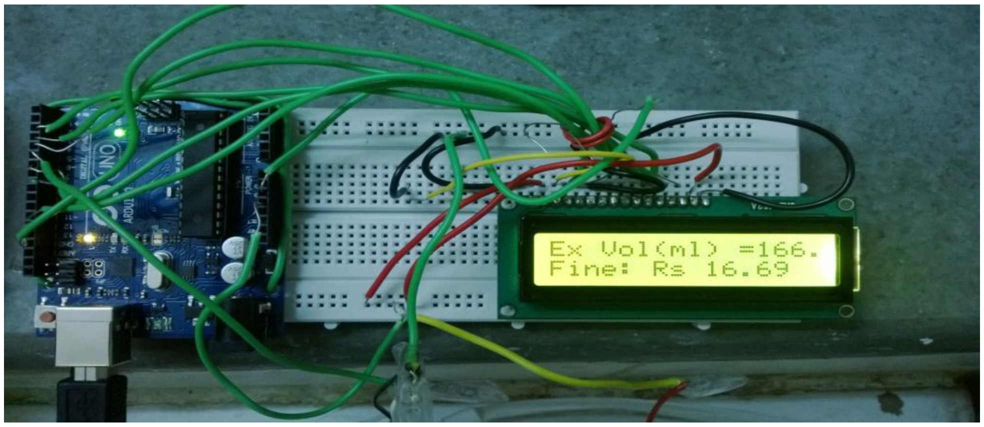


Fig 4.4 LCD showing the fine imposed

4.3 Fingerprint module

Result showing enrolment and verification of user using fingerprint module and interfacing the Hall Effect sensor and finger print module and then the user usage corresponding to its detection and verification.

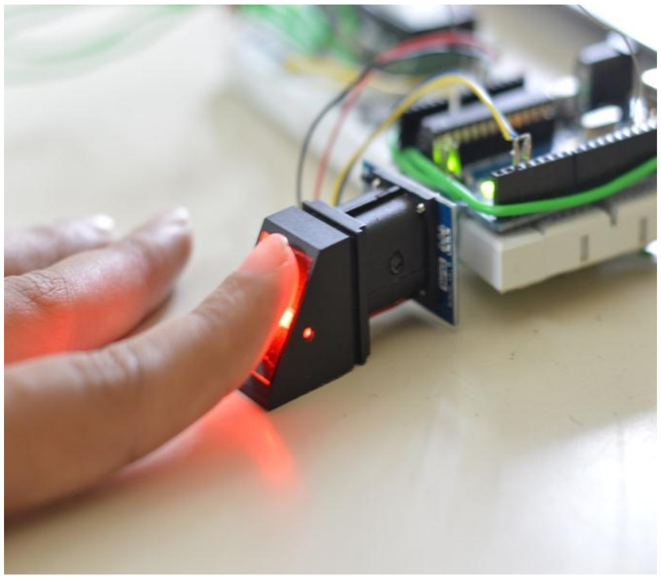
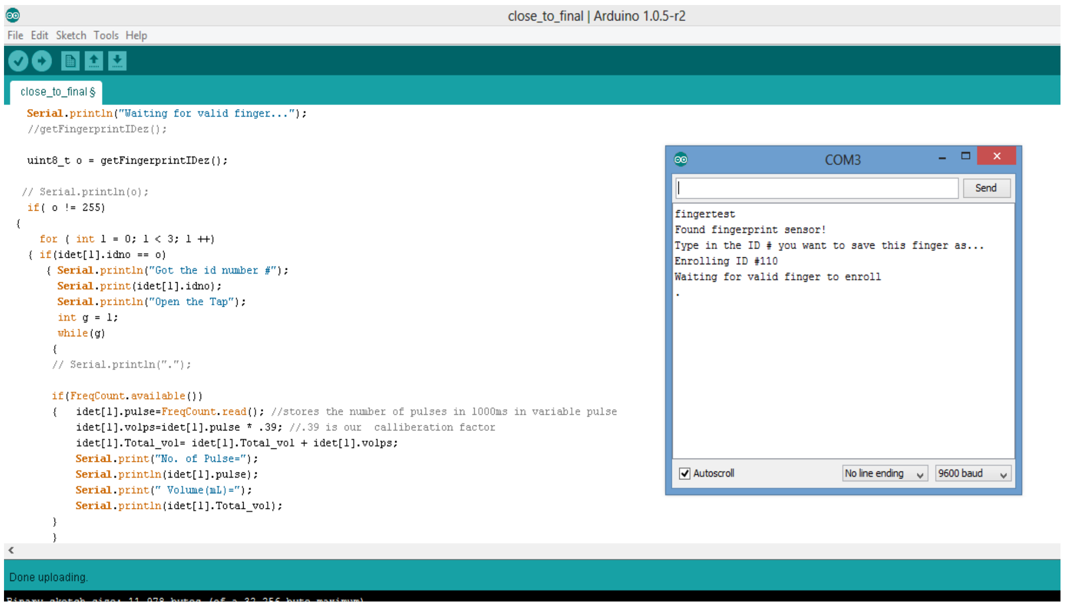


Fig 4.5 finger print module

Fig 4.5 shows the finger print module R305 we are using to identify the user before usage of water. We are concerned wih various aspect after identification as we identify the user its usage is also needs to be stored and the process in done using classes and object in Arduino programming where apart from displaying it we need to store it to use it for later purpose. Our challenge didn’t stop with the identification of the user but its usage is also of great concerned to us and it is the sole aim of the project. We in Fig 4.6 see that the module has been detected and is ready to be used. It is then followed by the enrolment process, result of which is shown in Fig 4.7

4.3.1 Enrolment and verification

The finger print module is been detected and similarly the the process of enrolment and verification is carried out and the process is encapsulated in the figure shown below



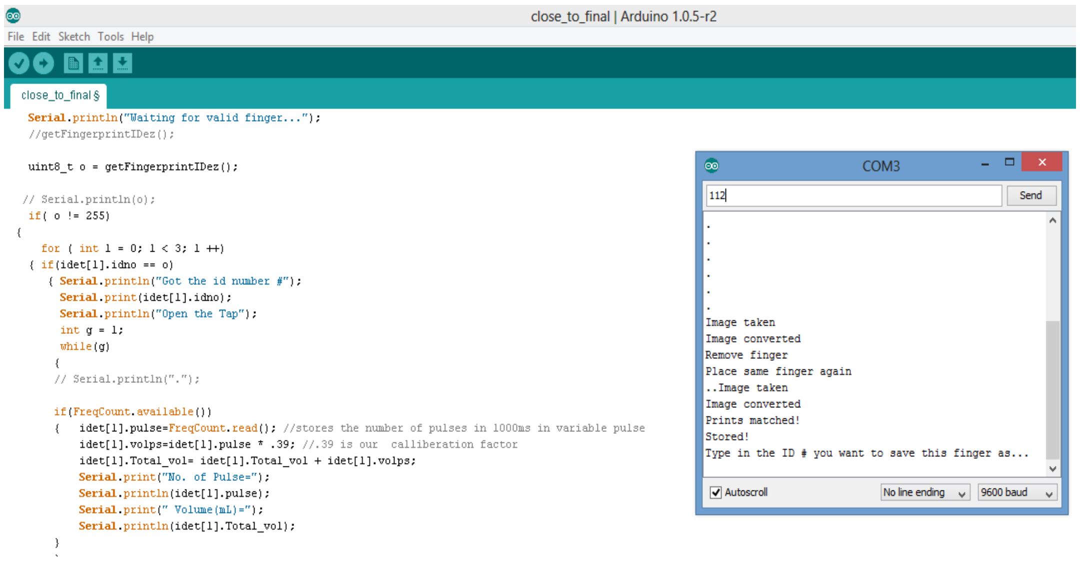
```
 Serial.println("Waiting for valid finger...");
 //getFingerprintIDez();

 uint8_t o = getFingerprintIDez();

// Serial.println(o);
 if( o != 255)
{
   for ( int l = 0; l < 3; l ++)
 { if(idet[l].idno == o)
   { Serial.println("Got the id number #");
    Serial.print(idet[l].idno);
    Serial.println("Open the Tap");
    int g = 1;
    while(g)
   {
   // Serial.println(".");

   if(FreqCount.available())
   {   idet[l].pulse=FreqCount.read(); //stores the number of pulses in 1000ms in variable pulse
       idet[l].volps=idet[l].pulse * .39; //.39 is our  calliberation factor
       idet[l].Total_vol= idet[l].Total_vol + idet[l].volps;
       Serial.print("No. of Pulse=");
       Serial.println(idet[l].pulse);
       Serial.print(" Volume(mL)=");
       Serial.println(idet[l].Total_vol);
   }
   }
```

Done uploading.

```
fingertest
Found fingerprint sensor!
Type in the ID # you want to save this finger as...
Enrolling ID #110
Waiting for valid finger to enroll
.
```

Fig 4.6 Finger print module detected

We here see the various aspect dicussed above and the result are positive as we are able to identify user by successfully enrolling and verifying shown in Fig 4.7 and Fig 4.8 respectively

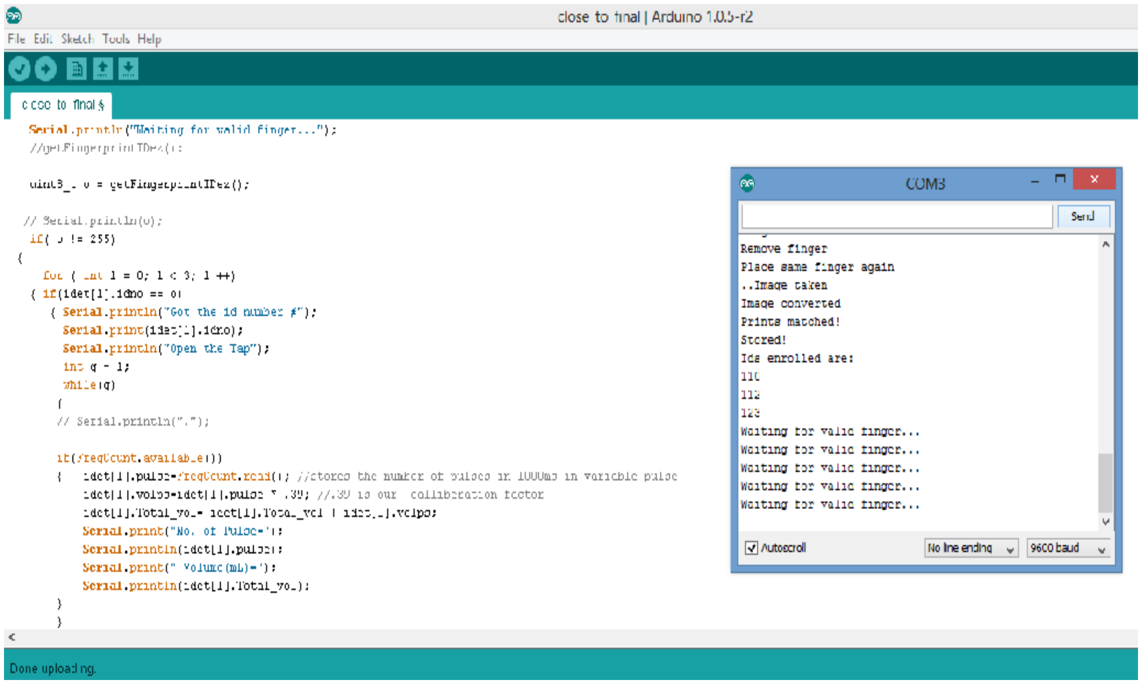


```
 Serial.println("Waiting for valid finger...");
 //getFingerprintIDez();

 uint8_t o = getFingerprintIDez();

// Serial.println(o);
 if( o != 255)
{
   for ( int l = 0; l < 3; l ++)
 { if(idet[l].idno == o)
   { Serial.println("Got the id number #");
    Serial.print(idet[l].idno);
    Serial.println("Open the Tap");
    int g = 1;
    while(g)
   {
   // Serial.println(".");

   if(FreqCount.available())
   {   idet[l].pulse=FreqCount.read(); //stores the number of pulses in 1000ms in variable pulse
       idet[l].volps=idet[l].pulse * .39; //.39 is our  calliberation factor
       idet[l].Total_vol= idet[l].Total_vol + idet[l].volps;
       Serial.print("No. of Pulse=");
       Serial.println(idet[l].pulse);
       Serial.print(" Volume(mL)=");
       Serial.println(idet[l].Total_vol);
   }
```

```
112
```

```
.
.
.
.
.
.
Image taken
Image converted
Remove finger
Place same finger again
..Image taken
Image converted
Prints matched!
Stored!
Type in the ID # you want to save this finger as...
```

Fig 4.7 Finger print stored



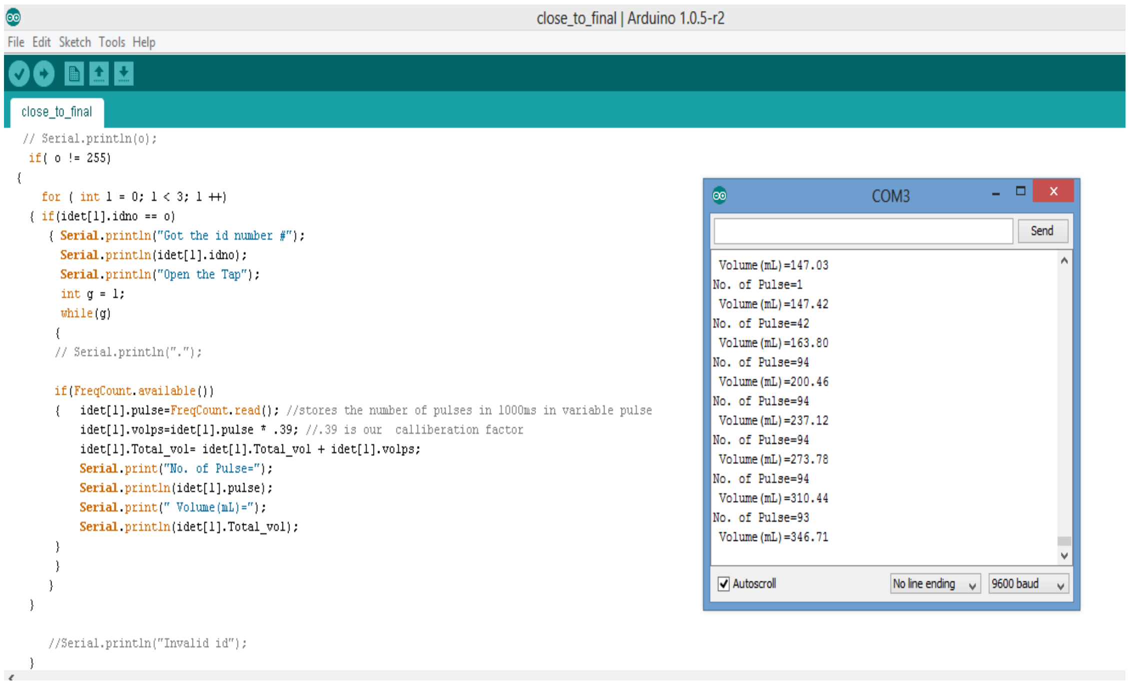
```
Serial.println("Waiting for valid finger...");
//getFingerprintIDez();

uint8_t o = getFingerprintIDez();

// Serial.println(o);
if( o != 255)
{
  for ( int l = 0; l < 3; l ++)
 { if(idet[l].idno == o)
   { Serial.println("Got the id number #");
     Serial.print(idet[l].idno);
     Serial.println("Open the Tap");
     int g = 1;
     while(g)
    {
    // Serial.println(".");

    if(FreqCount.available())
    {   idet[l].pulse=FreqCount.read(); //stores the number of pulses in 1000ms in variable pulse
        idet[l].volps=idet[l].pulse * .39; //.39 is our  calliberation factor
        idet[l].Total_vol= idet[l].Total_vol + idet[l].volps;
        Serial.print("No. of Pulse=");
        Serial.println(idet[l].pulse);
        Serial.print(" Volume(mL)=");
        Serial.println(idet[l].Total_vol);
    }
    }
```

Done uploading.

```
Remove finger
Place same finger again
..Image taken
Image converted
Prints matched!
Stored!
Ids enrolled are:
110
112
123
Waiting for valid finger...
Waiting for valid finger...
Waiting for valid finger...
Waiting for valid finger...
Waiting for valid finger...
```

close to final | Arduino 1.0.5-r2
File Edit Sketch Tools Help
close to final §
COM3
Send
Autoscroll
No line ending
9600 baud

Fig 4.8 Verification of the user



```
  // Serial.println(o);
  if( o != 255)
 {
   for ( int l = 0; l < 3; l ++)
  { if(idet[l].idno == o)
    { Serial.println("Got the id number #");
      Serial.println(idet[l].idno);
      Serial.println("Open the Tap");
      int g = 1;
      while(g)
     {
     // Serial.println(".");

     if(FreqCount.available())
     {   idet[l].pulse=FreqCount.read(); //stores the number of pulses in 1000ms in variable pulse
         idet[l].volps=idet[l].pulse * .39; //.39 is our  calliberation factor
         idet[l].Total_vol= idet[l].Total_vol + idet[l].volps;
         Serial.print("No. of Pulse=");
         Serial.println(idet[l].pulse);
         Serial.print(" Volume(mL)=");
         Serial.println(idet[l].Total_vol);
     }
     }
    }
 }

    //Serial.println("Invalid id");
 }
```

```
 Volume(mL)=147.03
No. of Pulse=1
 Volume(mL)=147.42
No. of Pulse=42
 Volume(mL)=163.80
No. of Pulse=94
 Volume(mL)=200.46
No. of Pulse=94
 Volume(mL)=237.12
No. of Pulse=94
 Volume(mL)=273.78
No. of Pulse=94
 Volume(mL)=310.44
No. of Pulse=93
 Volume(mL)=346.71
```

close_to_final | Arduino 1.0.5-r2
File Edit Sketch Tools Help
close_to_final
COM3
Send
Autoscroll
No line ending
9600 baud

Fig 4.9 Usage corresponding to the enrolled user usage

The final step shown in Fig 4.9 in which we are able to get all the data which is concerned with the usage of user and corresponding to which we stored all the values so that we get the data of the all the user and hence we can monitor water usage.

Chapter 5

CHALLENGES

We came across many problems while designing this system, some of the problems we have mentioned here with their solution.

Problem -

When we were to design this system using Arduino processor, we were newly introduced to the world of Arduino. Then the question came, what is Arduino processor?

Solution-

We took a survey of Arduino Processors and their features. We studied the Arduino User manual to know about Arduino processor.

Problem -

How to program Arduino processor? What is Dynamic C?

Solution-

Dynamic C is software provided to program Arduino processor. Whenever you v purchase Arduino kit then you will get this software downloaded from the web.

Problem -

How to write any general program in Dynamic C? How to execute it on Rabbit Processor?

Solution-

Then we studied the Dynamic C function manual to know meaning and syntax v of each and every function. Then we practiced some sample programs and tried to execute them on Arduino board kit.

Problem -

We only knew theory of Hall Effect and Hall Effect sensor, but how to attach it to on Arduino processor?

Solution-

We studied Hall Effect sensor data line from the manual provided and studied how to implement it on Arduino processor from jeremy blum blog on interfacing various devices to Arduino.

Problem -

After finishing all the programs, we were ready to execute them on Arduino Kit. But the necessary library files for them were missing. Hence, how to proceed?

Solution-

We searched on various forums site and posted queries related to non availability of library files of Arduino Processor. Mainly we posted the posts on the official site of Arduino.cc. We got help related to library files from various people all around the globe.

Problem -

After getting all the required library files, how to use them in the program and edit them as per requirement?

Solution-

Then also we posted queries on the forums while editing the library files as per the requirement. We got the help from forums only.

Problem-

How to find the ports for hardware interfacing?

Solution-

We searched into all the required library files to finds the ports for connection of Data and Control Signals.

Problem-

What finger print module are you using?

Solution-

R305 is the finger print module we are using which is easily available and affordable in India.

Problem-
What were the problem faced during interfacing of the finger print module with Arduino?

Solution-

Due to limited power supply available with Arduino and with two device already attached with Arduino we faced the problem of power supply and the problem was resolved by a external power source of 5V so that all the devices get proper power supply.

Problem-
Getting acquainted with the library function used in Arduino for finger print module?

Solution-

Library function were another hindrance and were properly studied in order to understand and to make the code for finger print module.

Problem-

How much feasible is the Industrial use of the model ?

Solution-

The sensor used does not match the efficiency required in an industry. Here we have used a cheaper sensor of small efficiency which must be replaced by a better one when the system is realized and implemented in an industry

Appendix I

REFERENCES

- Beginning Arduino- MICHAEL MCROBERTS

  A guide to understand the working of the Arduino board.

  Published by: Technology In Action

- http://Arduino.cc/en/Guide

- http://www.jeremyblum.com/category/Arduino-tutorials/software

  Basic understanding of usage of software and using various commands and library function

- http://www.youtube.com/watch?v=fCxzA9_kg6s

  Description about using Arduino initially. Getting acquainted with all the ports and their function with the environment provided to develop various project

- http://www.ele.uri.edu/courses/ele205/Arduino%20-%20Learning.pdf

  Getting acquainted with the serial communication used in Arduino and how we attach a sensor to the Arduino

- http://www.linuxjournal.com/content/learning-program-Arduino

  Understanding the basic knowledge of the programming of Arduino so as to use it in programming the project work

- http://www.instructables.com/id/Arduino-All-in-One-Getting-Started-Guide

  The interfacing of sensors and memory devices with the Arduino

- http://opensourcehardwaregroup.com/theArduinocourse

  The basic of LCD library operation and how to attach it with Arduino

- https://learn.adafruit.com/adafruit-optical-fingerprint-sensor/wiring-for-use-with-Arduino

  The theory of how to get acquainted with finger print module and and it programming, library function and interfacing with the Arduino.

- http://forum.Arduino.cc/index.php?topic=190205.g

  Basic forum on the programing of fingerprint module interfacing with the Arduino

- https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Due

  Fingerprint module basic.

- https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Library